Simulación de la Actividad Eléctrica del Corazón utilizando Unidades de Procesamiento Gráfico (GPU)

Autor: Andrés Mena Tobar
Director: José Félix Rodríguez

Máster en
Ingeniería Biomédica

Septiembre, 2012
Año Académico: 2011-2012

Universidad Zaragoza
1542

Simulación de la Actividad Eléctrica del Corazón utilizando Unidades de Procesamiento Gráfico (GPU)

Resumen

En los últimos años, las técnicas de observación celular y molecular han mejorado paulatinamente, lo que ha permitido avanzar en el conocimiento de los mecanismos que gobiernan la electrofisiología cardiaca a nivel celular y su relación con la señal bioeléctrica a nivel de tejido, órgano y superficie (electrocardiograma). En este sentido, la modelización matemática se ha convertido en una importante herramienta en la investigación de la biofísica celular y la electrofisiología. Los modelos matemáticos de células cardíacas pueden ser acopladas a modelos de tejido y ser empleados para simular la actividad eléctrica del corazón bajo condiciones normales y patológicas, así como bajo los efectos de medicamentos anti- y pro-arrítmicos. Uno de los objetivos de la modelización y simulación numérica es el de reproducir los experimentos, entender los fenómenos físicos involucrados que no pueden ser observados a través de ellos y el poder predecir esos fenómenos. Los modelos matemáticos de la electrofisiología del corazón describen de qué manera se propaga la onda eléctrica a través del tejido cardíaco y su relación con procesos que ocurren a escala celular. Los modelos involucrados en este trabajo, consisten en sistemas de ecuaciones diferenciales ordinarios (EDO) que modelan la electrofisiología de una célula, acoplados a un sistema de ecuaciones diferenciales en derivadas parciales (EDP) que gobiernan la propagación de la señal eléctrica a través del tejido.

El gran nivel de detalle electrofisológico de los modelos celulares subyacentes, convierte la simulación de la actividad eléctrica de un órgano como el corazón, en un desafío computacional de considerable envergadura. El reto desde el punto de vista computacional se encuentra en el correcto manejo de las diferentes escalas de tiempo y espacio existentes en el problema. Las constantes de tiempo involucradas en la cinética de los modelos iónicos van desde una fracción de milisegundo (corriente de sodio), hasta los cientos de milisegundos (corriente de calcio), lo cual implica desde un punto de vista numérico, pasos de integración del orden de centésimas de milisegundo. Por otro lado, la rápida despolarización de la membrana celular (el potencial de membrana varía 120mV en menos de un milisegundo), sumado a la relativamente baja velocidad de conducción del ventrículo (entorno a los 55 cm/s de velocidad media), implica que el frente de despolarización se desarrolla en pocos milímetros, dando lugar a la necesidad de emplear discretizaciones muy finas con la finalidad de obtener resultados fiables. De esta manera, la simulación de un solo latido cardíaco (800 ms de simulación), en una geometría real del corazón humano, conlleva resolver un problema con millones de grados de libertad y simulaciones que pueden requerir miles de pasos de tiempo. El objetivo principal de este trabajo es abordar la solución numérica del problema de electrofisología cardiaca en arquitecturas GPU (Graphic Processor Units) para cálculo de altas prestaciones aprovechando el alto nivel de paralelización de esta arquitectura. Para ello se implementó en C++ y CUDA un programa de elementos finitos para resolver el modelo mono-dominio para la simulación de la propagación de la actividad eléctrica en el corazón acoplado a modelos electrofisiológicos detallados.

Contenido

1 INTRODUCCION

1.1 Anatomía

El Corazón está ubicado en la cavidad torácica, en el mediastino medio, entre los dos pulmones e inmediatamente retro-esternal, es decir, tiene por delante el esternón y los cartílagos costales de la tercera, cuarta y quinta costillas, derechas e izquierdas. El tercio derecho del Corazón, apenas sobresale del borde esternal derecho y los dos tercios restantes, se sitúan a la izquierda, terminando en una punta (ver Figura 1.1)**.** Su cara inferior descansa, sobre el músculo Diafragma, que separa la cavidad torácica de la cavidad abdominal. Está orientado en el espacio, desde arriba hacia abajo, de derecha a izquierda y desde atrás hacia adelantes y el tamaño y peso, varían en forma considerable según la edad y sexo, pero en un adulto joven de estatura media, el corazón pesa entre 270 y 300 gramos [1].

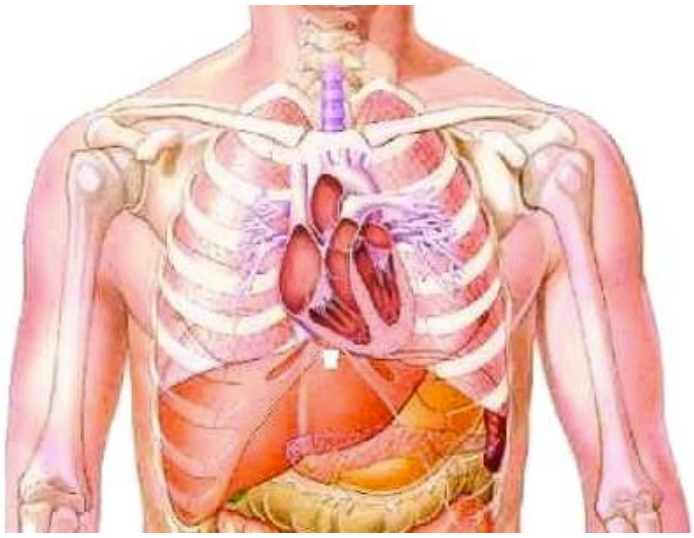


Figura 1.1: Localización anatómica del corazón.

El corazón está recubierto externamente por dos hojas de tejido seroso, llamadas Pericardio, una de ellas íntimamente adherida al órgano (epicardio) y otra que, continuándose con la primera, se refleja en la base en torno al corazón para rodearlo completamente (pericardio propiamente dicho); entre las dos hojas, que no están adheridas entre sí, existe una cavidad virtual que permite los libres movimientos de la contracción cardíaca. Está sostenido desde su parte superior por los grandes troncos arteriales. Estos son la arteria Aorta, arteria Pulmonar, Vena Cava Superior, Vena Cava Inferior y cuatro Venas Pulmonares.

1.1.1 Morfología interna

En su interior pueden observarse cuatro cavidades, dos superiores llamadas: aurícula derecha y aurícula izquierda, y dos inferiores, llamadas: ventrículo derecho, y ventrículo izquierdo respectivamente (ver Figura 1.2) [1]. Las aurículas están separadas entre sí por un tabique o Septum interauricular y los ventrículos por el Septum interventricular. Ambos tabiques se continúan uno con otro, formando una verdadera pared membranosa-muscular que separa el corazón en dos cavidades derechas y dos cavidades izquierdas. Esta separación es funcional, ya que las cavidades derechas se conectan con la circulación Pulmonar y las cavidades izquierdas, con la circulación general Sistémica.

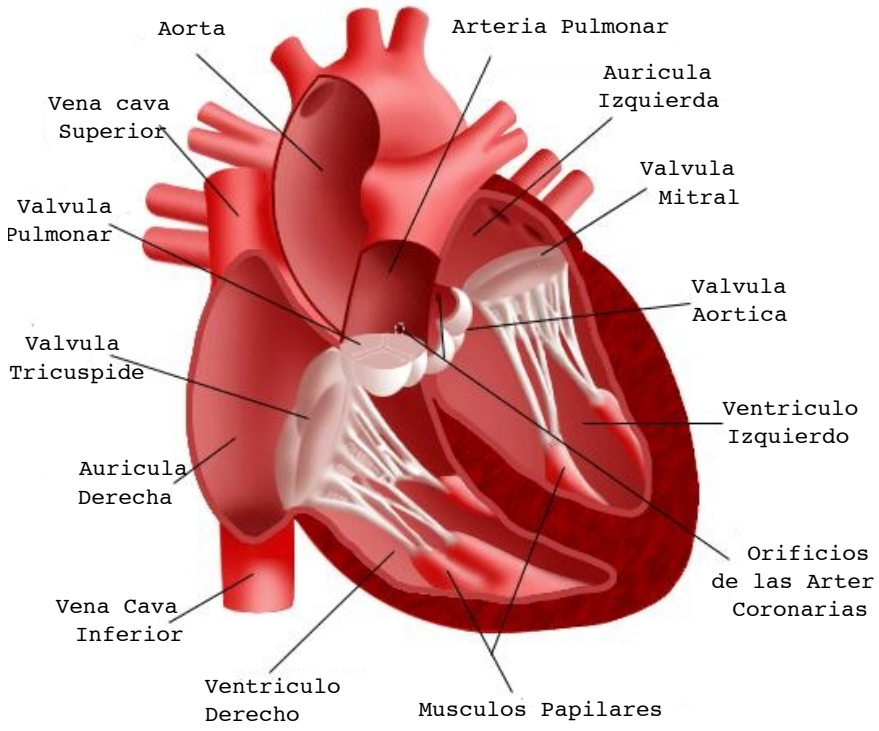


Figura 1.2: Principales Arterias, Venas y morfología interna.

A la aurícula derecha llegan las venas Cavas Superior e Inferior trayendo sangre carbo-oxigenada de todo el organismo. Pasa al ventrículo derecho, el cual al contraerse (Sístole Ventricular), la envía a la arteria Pulmonar que se dirige a ambos pulmones para efectuar el intercambio gaseoso. La sangre oxigenada regresa a la aurícula izquierda por medio de las cuatro venas pulmonares y ya en el ventrículo izquierdo, es bombeada hacia la arteria Aorta para ser distribuida por todo el organismo.

Las aurículas se comunican con los ventrículos a través de sendas válvulas, cuya función es abrirse para permitir el ingreso de sangre en la cavidad, luego de cerrarse herméticamente, durante la sístole para impedir que la misma refluya hacia atrás. Estas válvulas son la Mitral, entre aurícula y ventrículo izquierdos y la Tricúspide, entre aurícula y ventrículo derechos.

El ventrículo izquierdo vuelca su contenido a la arteria Aorta a través de la válvula Aórtica y el ventrículo derecho descarga a la arteria Pulmonar a través de la válvula Pulmonar. Ambas válvulas poseen tres valvas llamadas semilunares o sigmoideas formando una especie de estrella de tres puntas. A diferencia de las válvulas aurículo-ventriculares, estas carecen de cuerdas tendinosas que las sostenga y se cierran herméticamente ya que se parecen a diminutos paracaídas, que se abomban y contactan entre sí.

Anatómicamente el ventrículo derecho es delgado, ya que debe contraerse en contra de una presión muy baja, Su pared mide entre 4 y 5 MM. de espesor. El ventrículo izquierdo debe vencer la resistencia o presión arterial sistémica, por lo tanto su fuerza de contracción debe ser mayor. Por este motivo sus paredes son más gruesas, con un espesor de entre 8 y 15 mm.

El ciclo cardíaco se compone de tres fases:

1. Sístole auricular: fase en la cual las aurículas se contraen, se corresponde eléctricamente con la despolarización auricular.
2. Sístole ventricular: fase en la cual los ventrículos se contraen, se corresponde eléctricamente con la despolarización ventricular.
3. Diástole: Fase en la cual el músculo cardiaco se relaja, eléctricamente se corresponde con la repolarización ventricular.

1.2 Actividad eléctrica del corazón, estimulación y conducción.

Las células cardíacas, en su conjunto, generan sus propios estímulos para luego conducirlos rápidamente y activar los mecanismos contráctiles dentro de las mismas. Éstas constituyen el sistema de excitación y conducción del corazón, cuyo

funcionamiento explicaremos a continuación. La Figura 1.3 muestra el sistema de estimulación y conducción del corazón.

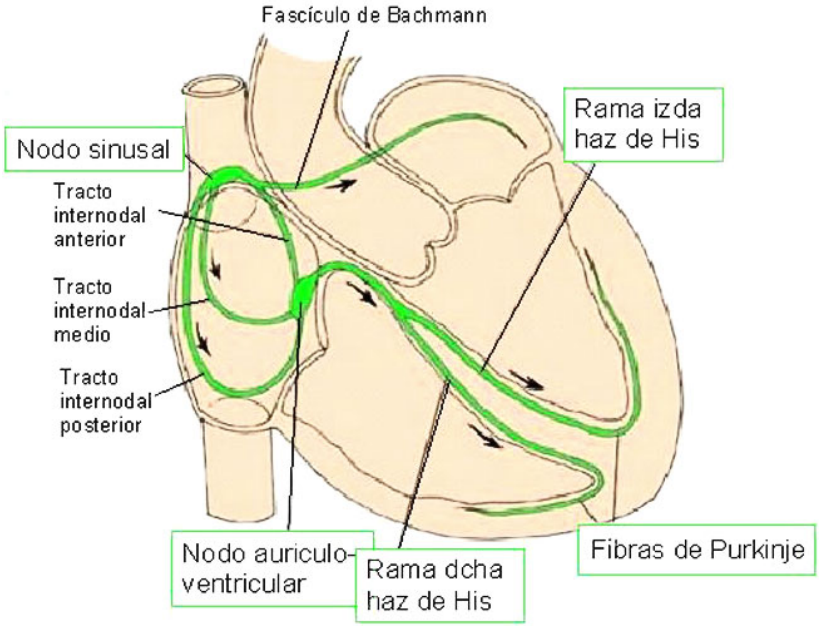


Figura 1.3 Sistema eléctrico del corazón, se muestra la totalidad del sistema desde el nodo SA hasta las arborizaciones de Purkinje.

Los estímulos cardíacos se originan en forma normal en el Nódulo Sinusal (SA), o de Keith y Flack (marcapasos normal de Corazón), ubicado en la unión de la Vena Cava superior con la aurícula derecha. El estímulo eléctrico generado viaja por los haces inter-nodales anterior, medio y posterior hasta el nódulo aurículo- ventricular (AV), o de Aschoff- Tawara, ubicado en el endocardio de la misma aurícula derecha pero en la región septal, inmediatamente por encima de la inserción de la valva septal de la válvula Tricúspide. El nódulo AV se prolonga en el ventrículo formando el Haz de His. El sector donde concurren los tres haces intermodales y el Haz de His se conoce como Unión Aurículo- Ventricular. El haz de His, desciende por el septo inter-ventricular, donde se divide en dos ramas principales: tronco de la Rama izquierda y tronco de la Rama Derecha. La rama derecha desciende por la cara derecha del tabique inter-ventricular, hasta la base del músculo papilar anterior, donde termina en una profusa ramificación o Red de Purkinje. El tronco de la rama izquierda atraviesa el septo y desciende por su cara izquierda, dividiéndose inmediatamente en una rama anterior que activa la cara antero-superior del ventrículo izquierdo. Esta rama está en

contacto con la válvula aórtica y es más larga y delgada que su compañera posterior. La rama posterior, mucho más corta y gruesa, termina en la cara posterior e inferior del Corazón. Todas las ramas terminan en la red de Purkinje, que se ubica en la región sub-endocárdica del miocardio inter-conectándose entre sí, de modo que la interrupción de la conducción de los estímulos por una de ellas, no impide la activación del músculo.

La velocidad de conducción del estímulo dentro de este sistema no es uniforme, aceptándose los valores promedios siguientes:

1. Fibras de Purkinje: 2 m/s.
2. Musculatura auricular: 0,8 a 1 m/s.
3. Musculatura ventricular: 0,5 a 0,7 m/s.
4. Nódulo AV: 0,2 m/s.

La baja velocidad de propagación en el nódulo AV parece establecida para proteger a los ventrículos de impulsos demasiado rápidos que pueden originarse en las aurículas.

1.2.1 Potencial de acción.

El potencial de acción (PA) es una señal bioeléctrica que corresponde a la diferencia de potencial entre el medio intracelular y extracelular, separados por la membrana celular [2].

La membrana además de separar los medios, contiene canales iónicos, intercambiadores y bombas electrogénicas que permiten la movilidad tanto pasiva como activa de iones entre ambos medios. Este intercambio de iones se produce, principalmente, gracias a la diferencia de concentraciones iónicas entre los dos medios y por tanto debido a las fuerzas de difusión y de campo eléctrico. Así, este flujo dinámico de iones a través de la membrana es el mecanismo responsable de la generación del potencial de acción. En la Figura 1.4 se representa el potencial de acción en células cardiacas y sus diferentes fases. También se pueden apreciar las diferentes corrientes que influyen en el mismo.

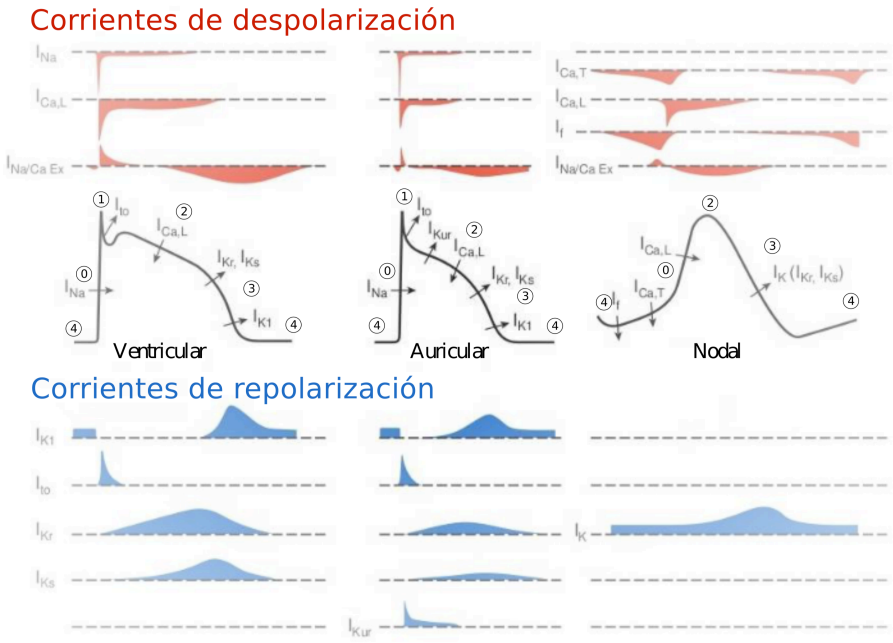


Figura 1.4: Principales corrientes que contribuyen al potencial de acción y fases del mismo. Imagen tomada de [2].

1.2.2 Fases del Potencial de Acción

El PA consta de varias fases (Fase 0, 1, 2, 3 y 4), las cuales se caracterizan por la activación de distintas corrientes iónicas. Las fases del PA se pueden describir brevemente como:

•**Fase 0**. Consiste en una fase inicial creciente del potencial de acción (aproximadamente 1 ms) y es lograda por la corriente de sodio (I_{na}) y la corriente diferida de entrada de calcio ($I_{ca(L)}$). La primera corriente en activarse es la I_{na}, cuyo elevado valor pico produce una rápida subida del potencial de membrana que permite la activación de la $I_{ca(L)}$, la cual seguirá activada durante el resto de la despolarización y durante toda la meseta.

•**Fase 1**. Después de que el potencial de acción ha alcanzado su valor pico, una rápida repolarización ocurre y se atribuye a la activación de las componentes de la corriente transitoria de salida de potasio (I_{to}). La magnitud de I_{to} varía marcadamente en diferentes localizaciones del corazón a la vez que es modulada dinámicamente por el ritmo cardíaco y otros factores. La densidad de I_{to} es más alta en miocitos epicárdicos ventriculares siendo la responsable la forma espiga-domo en la meseta del potencial.

•**Fase 2**. La fase lenta de repolarización ocupa la mayor parte de la meseta del potencial, y su dinámica está determinada por la inactivación de la corriente asociada a los canales de Calcio tipo L ($I_{ca,(L)}$) y la activación de las corrientes rectificadoras retardadas del potasio (I_k). La contribución relativa de I_{kr} e I_{ks} en la repolarización cambia según la especie y entre diferentes regiones celulares de la pared ventricular.

•**Fase 3**. Hacia el final de la meseta del potencial de acción la velocidad de repolarización se acelera de manera considerable debido al rápido incremento en las corrientes I_{kr} e I_{k1}. La salida de iones a través de estos canales se incrementa hasta crear una realimentación positiva que causa la rápida repolarización. La corriente I_{kr} controla la porción inicial de la repolarización mientras que I_{k1} controla la porción tardía.

•**Fase 4**. El potencial de membrana de la célula está en reposo, siendo éste similar al potencial de equilibrio del ión de potasios, y la corriente rectificadora inversa de potasio (I_{k1}) está activa.

1.3 Motivación y objetivos

En los últimos años, las técnicas de observación celular y molecular han mejorado paulatinamente, lo que ha permitido avanzar en el conocimiento de los mecanismos que gobiernan la electrofisiología cardiaca a nivel celular y su relación con la señal bioeléctrica a nivel de tejido, órgano y de superficie (electrocardiograma). En este sentido, la simulación por ordenador se está convirtiendo hoy día en una importantísima herramienta en la investigación de la biofísica celular y la electrofisiología.

Los modelos matemáticos de células cardíacas pueden ser acopladas a modelos de tejido y ser empleados para simular la actividad eléctrica del corazón bajo condiciones normales y patológicas, así como bajo los efectos de medicamentos. actualmente, el desarrollo de un medicamento cuesta millones de euros y las enfermedades cardiacas son una de las principales causas de muerte en todo el mundo.

Uno de los objetivos de la simulación es reproducir los experimentos, entender los fenómenos físicos involucrados que no pueden ser observados a través de ellos y lo más importante de todo es poder predecir esos fenómenos. Los modelos matemáticos de la electrofisiología del corazón describen de qué manera se propaga la onda eléctrica a través del tejido cardíaco. Los modelos involucrados en este trabajo, que serán descritos en detalle en el siguiente capítulo, consisten en sistemas de ecuaciones diferenciales ordinarios (EDO) que modelan la electrofisiología de una célula, acoplados a un sistema de ecuaciones diferenciales en derivadas parciales (EDP) que gobiernan la propagación de la señal eléctrica a través del tejido.

A medida que se dispone de más datos experimentales y de más información de los mecanismos que gobiernan la actividad electrofisiológica a nivel celular, esta información se incorpora a los modelos matemáticos, obteniéndose modelos fisiológicamente más reales, pero también más complejos desde un punto de vista matemático y más difíciles y costosos de resolver desde el punto de vista numérico.

El incremento en el detalle electrofisológico de los modelos celulares subyacentes, convierte la simulacion de la actividad eléctrica de un órgano como el corazón, en un desafío computacional de considerable envergadura. El reto desde el punto de vista computacional es debido principalmente a las diferentes escalas de tiempo y espacio existentes en el mismo. Las constantes de tiempo involucradas en la cinética de los modelos iónicos van desde una fracción de milisegundo (corriente de sodio), hasta los cientos de milisegundos (corriente de calcio), lo cual implica desde un punto de vista numérico, pasos de integración del orden de centésimas de milisegundo. Por otro lado, la rápida despolarización de la membrana celular (en un milisegundo el potencial de membrana varía 120mV) sumado a la relativamente baja velocidad de conducción del ventrículo (entorno a los 55 cm/s de velocidad media), implica que el frente de despolarización se desarrolla en pocos milímetros, dando lugar a la necesidad de emplear discretizaciones muy finas con la finalidad de obtener resultados fiables. De esta manera, la simulación de un solo latido cardíaco (800 ms de simulación), en una geometría real del corazón humano, puede conllevar trabajar con problemas con cerca del orden de 10^7 grados de libertad y simulaciones que pueden requerir miles de pasos de tiempo.

El objetivo principal de este trabajo es abordar la solución numérica del problema de electrofisología cardiaca en arquitecturas GPU (Graphic Processor Units) para cálculo de altas prestaciones. Para ello se implementará en C++ y CUDA un programa de elementos finitos para resolver el modelo mono-dominio para la simulación de la propagación de la actividad eléctrica en el corazón acoplado a modelos electrofisiológicos detallados.

2 Modelo matemático

2.1 Modelo de tejido

El modelo matemático que describe la variación del potencial intracelular, V_i, potencial extracelular, V_e, y el potencial transmembrana, $V = V_e\text{-}V_i$, viene dado por el conocido modelo bidominio [3,4]

$$\nabla \cdot (\mathbf{D_i} \nabla V) + \nabla \cdot (\mathbf{D_i} \nabla V_e) = C_m \frac{\partial V}{\partial t} + J_{ion}(V, \boldsymbol{u}) \tag{1}$$

$$\nabla \cdot (\mathbf{D_i} \nabla V) + \nabla \cdot \left((\mathbf{D_i} + \mathbf{D_e}) \nabla V_e \right) = 0 \tag{2}$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(V, \mathbf{u}, t) \tag{3}$$

donde $\mathbf{D_i}$, y $\mathbf{D_e}$ son tensores de segundo orden positivo definidos que definen la conductancia de los medios intra- y extra-celular, $J_{ion}(V, \mathbf{u})$ es la corriente iónica transmembrana, $\mathbf{u}(\mathbf{x},t)$ es un vector de variables de estado asociado a la apertura y cierre de los canales iónicos y a las concentraciones iónicas intracelulares, $\mathbf{f}$ es una función vectorial de variables vectoriales asociada al modelo de comportamiento celular, y t se refiere al tiempo. Ambos, J_{ion} y $\mathbf{f}$ dependen del modelo celular utilizado. Las condiciones de contorno asociados a este modelo son de corriente nula a travé de la superficie del órgano

$$\mathbf{n} \cdot (\mathbf{D}_\mathrm{i} \nabla \mathrm{V} + \mathbf{D}_\mathrm{i} \nabla \mathrm{V}_\mathrm{e}) = 0 \tag{4}$$

$$\mathbf{n} \cdot (\mathbf{D}_\mathrm{e} \nabla \mathrm{V}_\mathrm{e}) = 0 \tag{5}$$

donde $\mathbf{n}$ es la normal exterior.

El acoplamiento presente entre la ecuación parabólica (1) y la ecuación elíptica (2) hacen que el modelo bidominio sea, en general, costoso de resolver y analizar. Sin embargo, este modelo puede simplificarse a una sola ecuación en derivadas parciales (EDP) haciendo algunas simplificaciones referentes a los tensores de conductividad. Si suponemos $\mathbf{D_e} = \lambda\ \mathbf{D_i}$, el modelo bidominio se reduce al conocido modelo monodominio

$$\nabla \cdot (\mathbf{D} \nabla V) = C_m \frac{\partial V}{\partial t} + J_{ion}(V, \boldsymbol{u}) \tag{6}$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(V, \mathbf{u}, t) \tag{7}$$

con condiciones de contorno

$$\mathbf{n} \cdot (\mathbf{D}\nabla \mathrm{V}) = 0 \tag{8}$$

El tensor $\mathbf{D} = (\lambda/(\lambda + 1))\mathbf{D_i}$ en las ecuaciones (6-8) es el tensor de conductancia efectivo.

El modelo monodominio representa una simplificación importante del modelo bidominio con ventajas importantes para el análisis matemático y computacional, lo que resulta adecuado para estudiar el comportamiento eléctrico del corazón.

Desde un punto de vista matemático y computacional, el problema definido por (6-8) se corresponde con la solución de una ecuación diferencial parcial lineal, que describe la conducción eléctrica, acoplado con un sistema no lineal rígido de ecuaciones diferenciales ordinarias que describen las corrientes iónicas transmembrana dando lugar a un problema de reacción-difusión nolineal. Una manera eficiente de resolver las ecuaciones (6-8) es mediante la aplicación de la técnica de descomposición de operadores [5]. La técnica de descomposición del operadores se ha aplicado a las ecuaciones mono-dominio en [6,7]. Los pasos básicos se resumen a continuación.

- Paso 1: Utilización de *V(t)* como la condición inicial para integrar la ecuación

$$\nabla \cdot (\mathbf{D}\nabla V) = C_m \frac{\partial V}{\partial t}, \text{ para } t \in [t, t + \Delta t/2] \tag{9}$$

- Paso 2: Utilizar el resultado obtenido en el Paso 1 como las condiciones iniciales para integrar

$$\begin{aligned} C_m \frac{\partial V}{\partial t} &= -J_{ion}(V, \mathbf{u}), \\ \frac{\partial \mathbf{u}}{\partial t} &= \mathbf{f}(\mathbf{u}, V, t) \end{aligned} \quad \text{para } t \in [t, t + \Delta t] \tag{10}$$

- Paso 3: Utilizar el resultado obtenido en la Paso 2 como la condición inicial para integrar

$$\nabla \cdot (\mathbf{D}\nabla V) = C_m \frac{\partial V}{\partial t}, \text{ para } t \in [t + \Delta t/2\,, t + \Delta t] \tag{11}$$

En la práctica, los pasos 1 y 3 se pueden combinar en uno, a excepción del primer incremento. Por lo tanto, después del primer incremento, el algoritmo sólo tiene dos pasos, Paso I correspondiente a la integración de las ecuaciones diferenciales

ordinarias (Paso 2) y Paso II correspondiente a la integración de la ecuación parabólica homogénea (Paso 1 y 3).

Al realizar el Paso II, el dominio de cálculo debe ser discretizado en el espacio por una malla bien sea de elementos finitos o diferencias finitas para aproximar las variables dependientes del problema, V y $\mathbf{u}$ [8]. Por lo tanto, después de la discretización espacial, el sistema de EDPs (9) o (11), se puede escribir en notación matricial como

$$\mathbf{M}\dot{\mathbf{V}} + \mathbf{K}\mathbf{V} = \mathbf{0} \tag{12}$$

donde $\mathbf{M}$ es la matriz de masa asociado con $C_m \partial V/\partial t$, y $\mathbf{K}$ es la matriz de rigidez asociada con $\nabla \cdot (\mathbf{D}\nabla V)$. Estas matrices se obtienen mediante el ensamblaje individual de matrices elementales.

A la ecuación (12) se le llama una ecuación semidiscreta porque el tiempo se deja continuo. El algoritmo de Euler explícito puede ser utilizado para integrar (12) en el tiempo [9]. Sea $\mathbf{V}^k$ un vector del potencial transmembrana en cada punto nodal del dominio discretizado (malla) en el tiempo t^k, donde k es el índice del paso de tiempo. Entonces en el tiempo t^{k+1} podemos escribir

$$\mathbf{M}\frac{\mathbf{V}^{k+1}-\mathbf{V}^k}{\Delta t} + \mathbf{K}\mathbf{V}^k = \mathbf{0} \tag{13}$$

Cuando se utiliza el algoritmo de descomposición del operador para resolver el modelo de monodominio, las ecuaciones (6)-(8) se resuelven en dos pasos. En primer lugar, el modelo celular electrofisiológico

$$\mathbf{V}^* = \mathbf{V}^k - \Delta t\, J_{ion}(\mathbf{V}^k, \mathbf{u}^k) \tag{14}$$

se resuelve en cada punto de malla para obtener un vector de potenciales transmembrana intermedio $\mathbf{V}^*$ (Paso I). A pesar de que un esquema de integración explícito se ha utilizado en (14), cualquier algoritmo para la solución de EDOs, ya sea implícito o explícito, puede emplearse en el cálculo de $\mathbf{V}^*$. Con esta solución intermedia a la mano, la ecuación (13) se convierte

$$\mathbf{M}\frac{\mathbf{V}^{k+1}-\mathbf{V}^*}{\Delta t} = -\mathbf{K}\mathbf{V}^k \tag{15}$$

o alternativamente

$$\mathbf{M}\mathbf{V}^{k+1} = \mathbf{M}\mathbf{V}^{*} - \Delta t\, \mathbf{K}\mathbf{V}^{k} \tag{16}$$

La ecuación (16) se resuelve para todo el dominio para obtener $\mathbf{V}^{k+1}$ (Paso II). Por lo tanto, el algoritmo de base en el tiempo t^{k+1} puede resumirse como:

• **Paso I:** Utilizar $\mathbf{V}^{k}$ como la condición inicial para integrar la ecuación (14) para obtener $\mathbf{V}^{*}$.

• **Paso II**: Utilizar el resultado obtenido en el Paso II para resolver (16) y obtener $\mathbf{V}^{k+1}$.

2.2 Modelo celular

En este trabajo se ha empleado el modelo de ten Tusscher y Panfilov 2006 [10,11] para simular la actividad eléctrica de las células cardiacas humanas ($J_{ion}(V,\mathbf{u})$ en la ecuación (6)). El modelo de ten Tusscher es de los denominados *modelo de segunda generación* [referencia] y representa uno de los modelos más avanzados actualmente. El modelo tiene dos versiones. En la primera versión del año 2004 [10], el modelo contempla los tres tipos de células cardiacas (endocardio, mid-miocardio y epicardio), así como las principales corrientes presentes en el modelo LR-II [12,13]. La segunda versión del modelo corresponde al año 2006 [11] en la que se introduce una mejora en el modelo de la dinámica del Calcio mediante la introducción de un subespacio para describir la liberación de calcio inducida por calcio (CICR por su acrónimo en inglés). La Figura 2.1 muestra un esquema detallado del modelo de ten Tusscher y Panfilov 2006 (TP06).

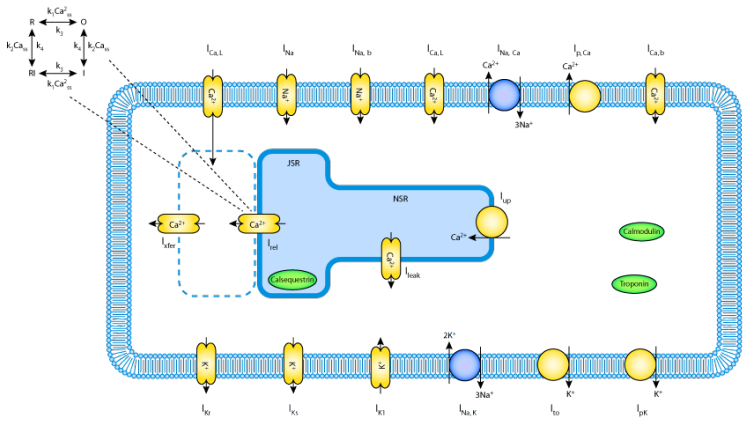


Figura 2.1. Esquema del modelo de ten Tusscher de 2006 [11].

La mayor novedad de la modificación del 2006, es la modificación de la dinámica del calcio y de la corriente de Calcio a través de los canales tipo L. En la dinámica del calcio se ha incluido un subespacio, SS, entre la membrana celular y el retículo sarcoplásmico, desde el cual se lleva a cabo la liberación de Calcio al citoplasma. Por su parte, la liberación del calcio desde el SR al SS se lleva a cabo a través de un mecanismo controlado por los receptores de Ryanodina cuya dinámica está gobernada por una cadena de Markov de cuatro estados. Sin embargo, en la implementación de 2006, la dimensionalidad de la cadena de Markov es reducida a dos estados con la finalidad de reducir el coste computacional del modelo. Con la introducción del SS, la corriente de Calcio a través de los canales tipo L inyecta Calcio en el SS, siendo ésta inactivada ahora por la concentración de calcio en el SS, $[Ca^{2+}]_{ss}$, en lugar de, $[Ca^{2+}]_i$. Adicionalmente, la nueva formulación incluye dos compuertas de inactivación dependientes de potencial, una lenta y una rápida. Desde el punto de vista computacional, el modelo TTP06 posee 19 variables de estado, 14 corrientes iónicas y un paso de integración mínimo de 0.02 ms.

3 Implementación del "solver" GPU

El modelo de programación CUDA (***C**ompute **U**nified **D**evice **A**rchitecture*) consiste en un programa maestro secuencial que puede ejecutar programas paralelos conocidos como "Kernels" en un dispositivo paralelo [14-16]. Un "Kernel" es a su vez, un programa que es ejecutado empleando un gran número de hilos paralelos. Cada hilo ejecuta el mismo código secuencial. En este sentido, el programador organiza los hilos de ejecución del "Kernel" en bloques de hilos y estos en mallas de bloques de ejecución. Los hilos pueden colaborar entre si, a través del uso de barreras de sincronización y compartiendo recursos de memoria en cada bloque, la cual es privada a cada uno de ellos. La Figura 3.1 muestra un esquema jerárquico del hilo, bloque y malla de ejecución con sus correspondientes espacios de memoria.

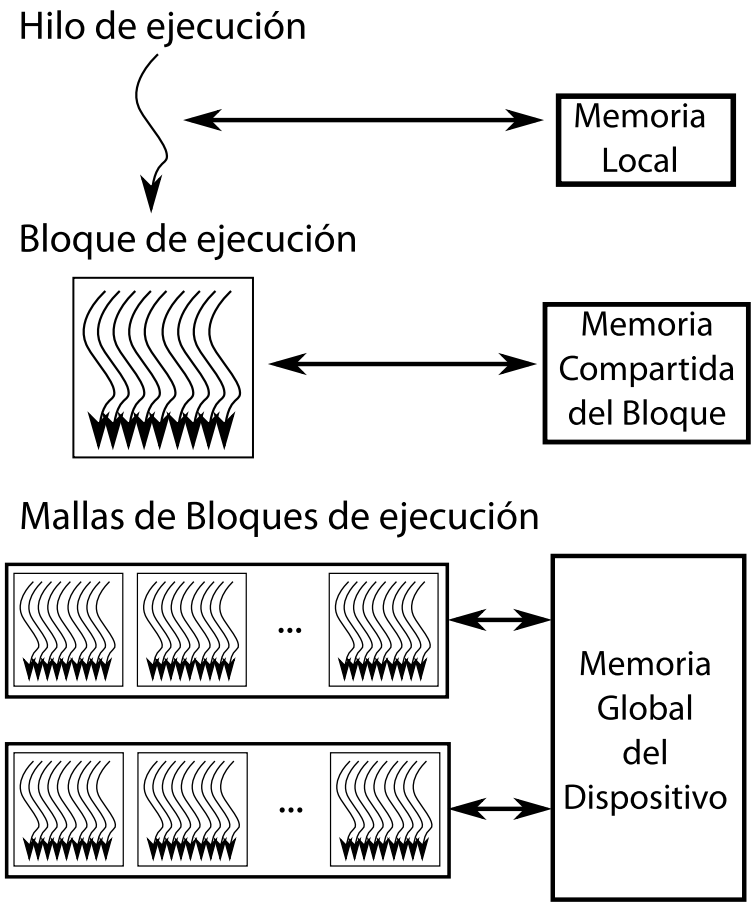


Figura 3.1. Estructura del vector de variables de estado del modelo iónico

Desde el punto de vista de "hardware", una GPU ejecuta una o más mallas de bloques; a su vez, cada bloque viene ejecutado a través de los llamados "Streaming Multiprocessors" (SM) que lanza los hilos en grupos de 32 denominados "warp". Una tarjeta GPU moderna, como la TESLA C2090 con tecnología FERMI, viene equipada de 16 multiprocesadores (SM) cada uno compuesto de 32 núcleos GPU para un total de 512 núcleos de procesamiento. La tarjeta tiene seis particiones de memoria de 64 bits, para una interfaz de memoria de 384 bits, soportando hasta un total de 6 GB memoria GDDR5 DRAM. La tarjeta está conectada a la CPU a través de una conexión PCI-Express. Las nueva tecnología mejora ostensiblemente el rendimiento en cálculo de punto flotante con doble precisión, lo cual ha permitido dar un salto cualitativo importante en la computación de alto rendimiento.

Hay dos aspectos esenciales que el programador ha de tomar en cuenta a la hora de implementar una aplicación en GPU: i) el orden de ejecución seguido por cada hilo dentro de un mismo warp, y ii) el orden de los datos en la memoria que es accedida por los hilos dentro del bloque. En lo que se refiere al orden de ejecución, un warp ejecuta una misma instrucción a través de todos sus hilos. A pesar que cada hilo dentro de un mismo warp puede seguir su propio orden de ejecución, es mucho más eficiente si cada hilo dentro del warp sigue el mismo orden de ejecución con la finalidad de garantizar que la computación se lleva a cabo en bloques. Esta característica debe ser considerada por el programador a la hora de escribir el código. El otro de los aspectos a considerar es el orden de los datos en la memoria del dispositivo. En este sentido, el acceso a datos no contiguos en memoria es perjudicial para el rendimiento ya que reduce el ancho de banda de memoria. Así, el programador debe garantizar la coalescencia de los datos en memoria para aquellos hilos dentro de un mismo bloque (i.e., los datos deben de estar en bloques continuos que pueden ser indexados por todos los hilos dentro del mismo bloque sin la necesidad de dar saltos).

La implementación en GPU del esquema numérico para la solución de las ecuaciones (14) y (16) se ha realizado empleando el método del elemento finito. La plataforma GPU elegida está compuestas de tarjetas fabricadas por nVIDIA™ por lo

que se utiliza el lenguaje de programación C++ y CUDA. **CUDA** son las siglas de ***C**ompute **U**nified **D**evice **A**rchitecture* (Arquitectura de Dispositivos de Cómputo Unificado) que hace referencia tanto a un compilador como a un conjunto de herramientas de desarrollo creadas por nVIDIA que permiten a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en GPU de nVIDIA [16]. CUDA intenta explotar las ventajas de las GPU frente a las CPU de propósito general utilizando el paralelismo que ofrecen sus múltiples núcleos, que permiten el lanzamiento de un altísimo número de hilos de ejecución simultáneos. Por ello, si una aplicación está diseñada utilizando numerosos hilos que realizan tareas independientes (que es lo que hacen las GPU al procesar gráficos, su tarea natural), una GPU podrá ofrecer un gran rendimiento en campos que podrían ir desde la biología computacional a la criptografía por ejemplo.

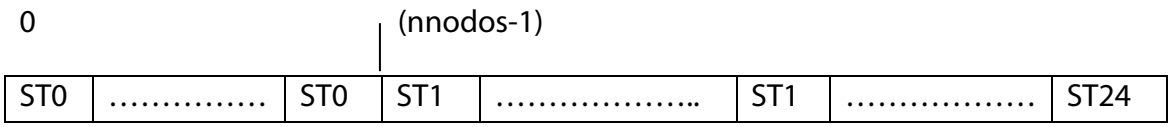
A continuación se procede a la explicación de la implementación del esquema numérico descrito en el capítulo anterior. La sección 3.1 está dedicada a la implementación de la integración del modelo iónico (Paso I), el modelo TP06 en nuestro caso. La sección 3.2 describe la implementación del método del elemento finito para resolver la ecuación (16), Paso II del algoritmo descrito en el capítulo anterior.

3.1. Implementación del módelo de TP06 en CUDA. "Solver" 0d

Primeramente, se realizó una fase de análisis del algoritmo, observándose que su coste computacional es demasiado alto, en uso de registros y operaciones, para que sea ejecutado por un solo hilo de ejecución. Por lo que hubo que dividir el procedimiento de cálculo del modelo iónico en varios subrutinas (kernels) de menor coste computacional.

En la implementación se ha empleado un vector de constantes con los parámetros del modelo iónico considerado al que acceden todos los hilos de ejecución. Se trata, por tanto, de un recurso compartido para todos los nodos en los que se ha definido el mismo modelo iónico. Asimismo, para garantizar un eficiente acceso de los datos por parte de todos los hilos dentro de un bloque, se emplea un

único vector que almacena las variables de estado del modelo iónico (STATES) para todos los nodos de la malla tal y como se muestra en la Figura 3.2.



ST0 => V	4 => Xr1	8 => h	12 => f
ST1 => Ki	5 => Xr2	9 => j	13 => f2
2 => Nai	6 => Xs	10 => Cass	14 => fCass
3 => Cai	7 => m	11 => d	15 => s
16 => r	17 => CasR	18 => RR	

Figura 3.2. Estructura del vector de variables de estado del modelo iónico

Note que esta estructura de datos garantiza que todos los hilos accederán a bloques continuos de datos garantizando la coalescencia de los mismos en memoria. Sin embargo, debido al límite en el número de instrucciones máximo que puede manejar cada hilo de ejecución, ha sido necesario duplicar seis de las variables de estado: V, Ki, Nai, Cai, Cass, y CasR, para un total de 24 con respecto a las 18 originalmente presentes en el modelo TP06, lo que implica un ligero incremento en el uso de memoria.

3.2 Implemenentación del modelo de propagación.

La solución de la ecuación (16) ha sido llevada a cabo empleando el método del elemento finito para poder resolver problemas 1D, 2D y 3D. El método del elemento finito es un método numérico para encontrar la solución a una ecuación o sistema de ecuaciones diferenciales ordinarias o en derivadas parciales como es el caso del modelo monodominio (6).

Uno de los pasos más importantes en la implementación del método del elemento finito consiste en la discretización del dominio (e.g., el corazón) en un número suficientemente elevado de subdominios no-intersectantes entre si, denominados *elementos finitos*. A la partición del dominio por los elementos finitos también se le denomina discretización. Los vértices de los elementos finitos de la

discretización se denominan *nodos*. Dos nodos se dicen adyacentes si pertenecen al mismo elemento, pudiendo un mismo nodo pertenecer a varios elementos. Al conjunto de nodos junto a la relación de adyacencia o conectividad define la malla sobre la que se resolverá la ecuación considerada. Una malla puede estar compuesta por diferentes tipos de elementos finitos. La Figura 3.4 muestra la tipología de elementos implementados en el "solver"

Debido a la variedad de elementos, se opta por una almacenamiento tipo *SPARSE* para el almacenamiento de la malla (descrito en la sección siguiente). De este modo el acceso a los elementos de la malla será siempre igual independiente del tipo. El formato utilizado es el CSR (compressed sparse row).

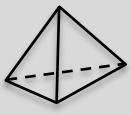
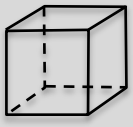
	ID_ELEMENTO	Nodos por elemento	Dimensión	Grados de libertad por elemento
	1	2	1D	2
	2	3	2D	3
	3	4	2D	4
	4	4	3D	4
	5	6	3D	8

Figura 3.3. Elementos finitos implementados en el "solver" CUDA

Las secciones siguientes describen el proceso de almacenaje y ensamblaje de las matrices de rigidez $\mathbf{K}$ y de masa $\mathbf{M}$ resultantes de la discretización del modelo monodominio presentes en (6). La siguiente sección detalla el ensamblaje de dichas matrices en la implementación del código en CUDA.

3.2.1 Ensamblaje de la matriz de rigidez y de masa.

Cuando se emplean elementos finitos para llevar a cabo la discretización espacial del problema, el potencial transmembrana, V^k, en el tiempo t^k, se aproxima a nivel elemental (en cada subdominio de la discretización) como

$$V^k = \sum_{j=1}^{N_{pe}} N_j (V_j^e)^k = \mathbf{N}^T (\mathbf{V}^e)^k, \tag{17}$$

donde N_j es la función de forma del nodo j del elemento e, $(V_j^e)^k$, es el valor del potnecial en el nodo j del elemento e en el tiempo t^k, y N_{pe} es el número de nodos por elemento [9]. Siguiendo la formulación de elmentos finitos estándar, las matrices de rigidez y de masa elementales de la ecuación vienen dadas como:

$$\begin{aligned} \mathbf{K}^e &= \int_{\Omega_e} \nabla \mathbf{N}^T \mathbf{D} \nabla \mathbf{N} \mathrm{d}\mathbf{x} \\ \mathbf{M}^e &= \int_{\Omega_e} C_{\boldsymbol{m}} \mathbf{N}^T \mathbf{N} \mathrm{d}\mathbf{x} \end{aligned} \tag{18}$$

donde Ω_e corresponde al subdominio definido por el elemento e.

La matriz de rigidez $\mathbf{K}$ asociada al sistema global (16) se obtiene por adición directa de matrices de rigidez elementales.

$$\mathbf{K} = \sum_{i=1}^{N_e} \mathbf{K}_i^e, \tag{19}$$

donde N_e corresponde al número total de elementos presentes en la discretización.

La matriz de rigidez $\mathbf{K}$ es una matriz del tipo dispersa (*matriz que tiene gran cantidad de ceros*), por lo que se almacena en formato SPARSE, evitando de este modo operaciones sobre los elementos que son cero (ahorrando tiempo de computación) y no guardando los elementos nulos (ahorro de memoria). Este formato hace uso de un vector de reales, $\mathbf{KA}$, de dimensión *n*, que contiene los elementos no nulos de la matriz $\mathbf{K}$ ordenados por filas (la dimensión n puede obtenerse a partir de la información de adyacencias de la malla), un vector $\mathbf{JA}$, también de dimensión *n*, que contiene los números de las columnas de los elementos de $\mathbf{K}$, e un vector $\mathbf{IA}$, de dimensión *nrow+1* que indica la cantidad de elementos no-nulos en cada una de las filas de $\mathbf{K}$. La Figura 3.5 muestra la estructura de almacenaje SPARSE para una matriz de 3x3

$$\mathbf{K} = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 5 & 0 \\ 0 & 0 & 6 \end{bmatrix} \qquad \begin{aligned} \mathbf{IA} &= [1 \quad 3 \quad 5 \quad 6] \\ \mathbf{JA} &= [1 \quad 2 \quad 1 \quad 2 \quad 3] \\ \mathbf{KA} &= [1 \quad 2 \quad 2 \quad 5 \quad 6] \end{aligned}$$

Figura 3.5. Matriz almacenada en formato SPARSE.

En la implementación sobre GPU, cada una de las matrices de rigidez elemental, $\mathbf{K}_i^e$, es calculada por un hilo de ejecución, mientras que el ensamblaje de la matriz global $\mathbf{K}$ tiene lugar a nivel de memoria global del CPU.

Una de las claves de una implementación computacionalmente eficiente de la ecuación (16) consiste en la integración de la matriz de masa $\mathbf{M}$ (18b). En lo que sigue, y sin pérdida de generalidad, se supondrá que $C_m=1.0$.

Cuando las funciones de forma, $\mathbf{N}$, empleadas para el cálculo de la matriz de masa $\mathbf{M}^e$ son las mismas que las empleadas para aproximar el potencial $\mathbf{V}$, el resultado es una matriz llena denomina matriz de masa *consistente*. En elementos finitos, la matriz de masa consistente no es una M-matriz (i.e, dada una fila i, la suma de los elementos fuera de la diagonal de la fila no es menor al elemento de la diagonal de dicha fila) lo cual puede dar origen a soluciones espurias en algunos problemas transitorios [17]. Por otro lado, el tener una matriz de masa llena, encarece sustancialmente la solución del sistema de ecuaciones (16) para actualizar el potencial $\mathbf{V}$, ya que obliga a la solución de un sistema de ecuaciones en cada incremento. Una alternativa para superar esta limitación consiste en diagonalizar la matriz de masa. Para diagonalizar la matriz de masa existen tres procedimientos bien establecidos: i) suma de filas, ii) escaldo de la diagonal, iii) empleo de quadratura nodal. En cualquier caso, el método de diagonalización debe de ser conservativo

$$\sum_i \widetilde{M}_{ii}^e = \int_{\Omega_e} d\mathbf{x}, \tag{20}$$

donde $\widetilde{\mathbf{M}}^e$ es la matriz de masa diagonalizada. La cuadratura nodal se basa en el uso de funciones de forma diferentes a las empleadas para aproximar el potencial de membrana V. Ya que no existen derivadas en la definición de $\mathbf{M}^e$, las funciones de forman pueden ser continuas a trozos dentro y entre elementos. Si por ejemplo las funciones de forma son constantes, tal que $\mathbf{N}_i=N_i$, en una parte del elemento

alrrededor del nodo *i*, y cero en el resto y además estas funciones son disjuntas (no se sobreponen), entonces la matriz de masa es diagonal

$$\int_{\Omega_e} N_i N_j \, d\mathbf{x} = \begin{cases} \int_{\Omega_e^i} d\mathbf{x} & i = j \\ 0 & i \neq j \end{cases} \tag{21}$$

Esta aproximación es admisible ya que satisface el criterio de integrabilidad y competitud [17]. En la implementación GPU considerada en este trabajo, se ha considerado una cuadratura nodal en la que $N_i = J_i$, siendo J_i el jacobiano elemental evaluado en el nodo *i* del elemento [17].

Al igual que para el caso de la matriz de rigidez, la matriz de masa global se obtiene como suma directa de las matrices elementales. En la implementación, la matriz de masa de cada elemento es calculada por un hilo de ejecución en la GPU, mientras que el ensamblaje se lleva a cabo a nivel de memoria global en el CPU. De esta manera, al finalizar el proceso de ensamblaje, tanto la matriz de masa como la matriz de rigidez se encuentran disponibles en la memoria de la CPU.

Una de las grandes ventajas computacionales de la diagonalización de la matriz de masa es que se evita la necesidad de resolver un sistema de ecuaciones en la actualización de V^{k+1} en (16), reduciendose a una simple multiplicación de matrices. Para la multiplicación de la matrices SPARSE, se ha empleado las funciones de librería de algebra lineal puestas a disposición por nVIDIA a través de la librería CUSP [18].

4 Resultados

Este Capítulo presenta la validación del modelo iónico implementado en GPU ("Solver0d"), así como de la implementación del modelo monodominio (Eq. 16). Todas las pruebas presentadas han sido realizadas en una tarjeta Tesla C2090 con tecnología fermi, con 6GB GDDR5 DRAM y 512 núcleos de procesamiento.

4.1. Validación de la implementación del modelo iónico

La validación del "solver" 0D consistió en estimular con una corriente de estímulo de -50pA/pF de 1ms de duración a una frecuencia de 1Hz durante 100s, todas las células del dominio de manera independiente, obteniendo el potencial de acción en cada una de ellas. El paso de integración empleado fue de 0.02ms. El potencial de acción y demás variables de estado correspondientes al último estímulo fueron comparadas con el resultado obtenido con la implementación original del modelo TP06 en C++, disponible en la página web de los autores[1]. La Figura 4.1 muestra los potenciales de acción obtenidos con el código C++ original y el calculado con la implementación de GPU.

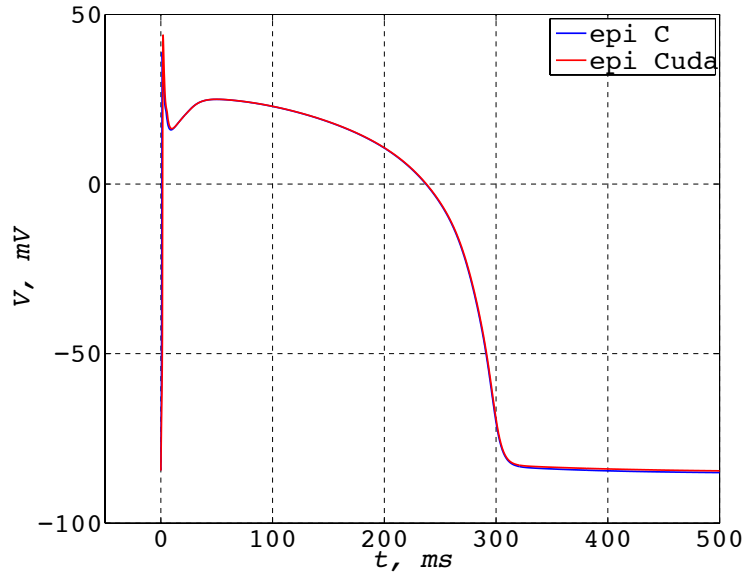


Figura 4.1 Potencial de acción obtenido con el código C++ originalmente del modelo TP06 (línea azul) y la implementación realizada en CUDA (línea roja).

Tal y como se puede observar en la figura, la correspondencia entre ambos resultados es excelente. Sin embargo, con la finalidad de cuantificar la calidad de los resultados

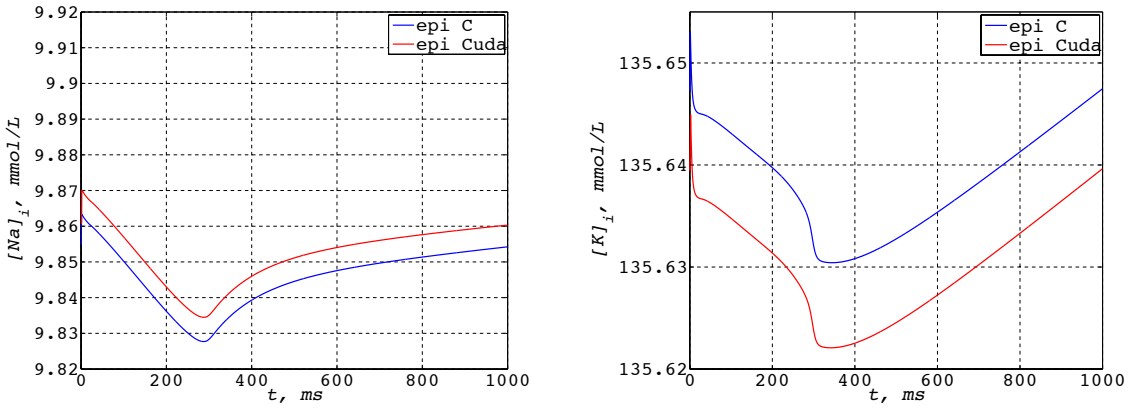
[1] http://www-binf.bio.uu.nl/khwjtuss/SourceCodes/

de la implementación, se calculó el coeficiente de determinación, R^2, de todas las variables de estado del modelo correspondientes al último estímulo del protocolo (ver Tabla 4.1).

Variable de Estado	R^2
V	0.9974
$[Ca]_i$	0.9993
$[Ca]_{SR}$	0.9982
$[Ca]_{SS}$	0.9388
$[Na]_i$	0.3691
$[K]_i$	-1.4931
m	0.9981
h	0.9967
j	0.9961
xs	1.0000
r	0.9971
s	0.9997
d	0.9976
f	1.0000
f2	0.9998
fcass	0.9973
R	0.9993
xr1	0.9997
xr2	0.9995

Tabla 4.1. Comparación de las resultados obtenidos con la implementación C++ original y la implementación en CUDA. El coeficiente de determinación ha sido calculado entre las curvas correspondientes al último estímulo del protocolo.

Los resultados de la Tabla 4.1 muestran que la correspondencia es excelente, a excepción de las concentración intracelular de Sodio y Potasio. Los valores de R^2 para estas dos variables de estado indican la existencia de un sesgo en estas variables tal y como se muestra en la Figura 4.2. Sin embargo, la diferencia es ,muy pequeña, apreciable tan solo en el tercer decimal y por tanto despreciables desde el punto de vista de la precisión de los resultados.



(a) (b)

Figura 4.2. Concentración intracelular de sodio [Na]i y de potasio [K]i obtenidas con el código C++ original y la implementación en GPU mostrando el sesgo existente. La diferencia es, sin embargo, solo apreciable en el tercer decimal de la variable.

4.2. Validación de la implementación en elementos finitos

Para validar la implementación en elementos finitos, se adoptó el protocolo propuesto en [20]. En este protocolo se considera un cuboide con dimensiones $3\times7\times20$ mm^3 tal y como se muestra en la Figura 4.3. El tejido se considera transversalmente isótropo con las fibras orientadas a lo largo del eje longitudinal del cuboide (20 mm). La geometría y dimensiones del problema lo convierten en un problema tratable del punto de vista numérico, así como fácilmente transferible y comparable en término de los resultados obtenidos, lo cual facilita la comparación entre diferentes códigos de simulación.

En lo referente a las condiciones de contorno, se ha considerado que todas las las caras del cubo satisfacen la condición de cero flujo (ecuación 8). La activación del tejido se lleva a cabo a través de la estimulación de un volumen de $1.5\times1.5\times1.5$ mm^3 localizado en una de las esquinas del cuboide (ver Figura 4.3).

Con la finalidad de caracterizar la convergencia en espacio y tiempo se han considerado tres discretizaciones espaciales ($\Delta x = 0.1, 0.2\ y\ 0.5$ mm) y tres discretizaciones temporales ($\Delta t = 0.005, 0.01\ y\ 0.05$ ms) resultando en un total de nueve experimentos numéricos. El cuboide fue discretizado empleando elementos hexaédricos regulares de tamaño de $\Delta x\times\Delta x\times\Delta x$.

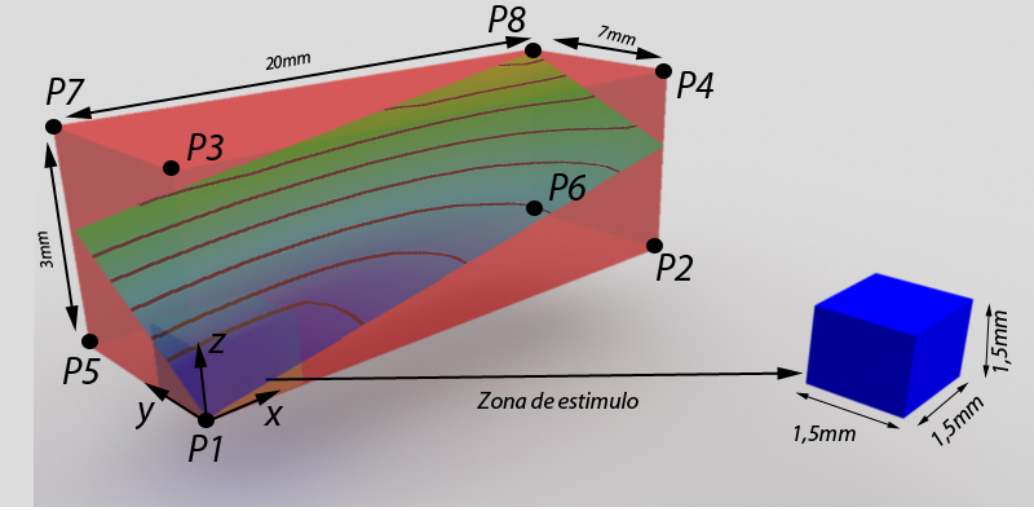


Figura 4.3. Geometría del dominio computacional considerado en el problema de validación. La estimulación se aplicó al cubo sombreado en la figura. El tiempo de activación fue medido en los puntos P1 al P8.

Las condiciones iniciales del modelo iónico se muestran en la Tabla 4.2

Variable de Estado	valor (unidades)	Variable de Estado	valor (unidades)
V	-85.23 (mV)	r	2.42×10^{-8}
$[Ca]_i$	0.000126 (mmolar)	s	0.999998
$[Ca]_{SR}$	3.64 (mmolar)	d	3.375×10^{-5}
$[Ca]_{SS}$	0.00036 (mmolar)	f	0.7888
$[Na]_i$	8.604 (mmolar)	f2	0.9755
$[K]_i$	136.89 (mmolar)	fcass	0.9953
m	0.00172	R	
h	0.7444	xr1	0.00621
j	0.7045	xr2	0.4712
xs	0.0095		

Tabla 4.2. Condiciones iniciales del modelo iónico

La validación del código se ha llevado a cabo a través de la comparación de los tiempos de activación en ocho puntos concretos del tejido y una superficie que secciona el tejido a lo largo de una de las diagonales mayores (ver Figura 4.3). El

tiempo de activación se define como aquel para el cual el potencial de membrana pasa por 0 mV.

La Figura 4.4 muestra el tiempo de activación en el punto P8 para las diferentes discretizaciones espacio-temporales consideradas. Cabe destacar que este punto representa una acumulación de error a medida que la onda viaja a través del tejido. También se muestran los tiempos de activación correspondientes a otros dos códigos de simulación implícitos basados en elementos finitos y MPI para paralelización: FEniCS y EMOS, desarrollado por el grupo SIMULIA y la Universidad de Zaragoza respectivamente. Como se puede observar en la figura, el desempeño de la implementación GPU en lo referente a precisión de la solución se encuentra entre las soluciones encontradas por los otros dos códigos.

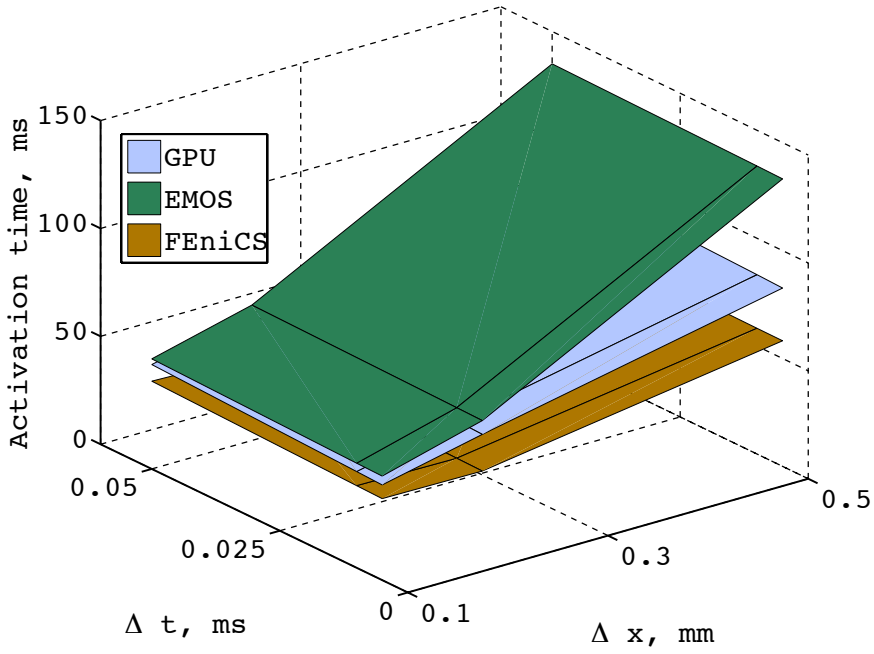


Figura 4.4. Tiempo de activación en el punto P8 (ver Figura 4.3) para las diferente discretizaciones espacio-temporales para la implementación GPU y dos códigos de simulación basados en MPI.

La Tabla 4.3 muestra el error en el tiempo de activación del punto P8 entre los tres códigos, para las diferentes discretizaciones espacio-temporales consideradas, así como el error cometido por cada código contra la solución real de 42.5 ms [20].

	Δt	0.005			0.01			0.05		
Δx	Código	GPU	FEniCS	EMOS	GPU	FEniCS	EMOS	GPU	FEniCS	EMOS
0.1	GPU	1.63	6.36	4.13	2.10	6.53	3.90	5.87	7.68	2.63
	FEniCS	6.36	4.73	10.48	6.53	4.43	10.43	7.68	1.80	10.30
	EMOS	4.13	10.48	5.75	3.90	10.43	6.00	2.63	10.30	8.50
0.2	GPU	11.88	16.98	6.38	12.38	17.29	6.13	16.13	19.38	4.13
	FEniCS	16.98	5.10	23.35	17.29	4.91	23.41	19.38	3.25	23.5
	EMOS	6.38	23.35	18.25	6.13	23.42	18.50	4.13	23.5	20.25
0.5	GPU	40.13	24.43	50.63	40.63	24.85	50.38	43.63	26.98	48.88
	FEniCS	24.43	15.78	75.05	24.85	15.78	75.22	26.98	15.78	75.85
	EMOS	50.63	75.05	91.00	50.38	75.22	91.00	48.88	75.85	92.50

Tabla 4.3. Diferencia en el tiempo de activación para el punto P8 entre los diferentes códigos. Los número en rojo indican la diferencia en el tiempo de activación con la solución real de 42.5ms [20].

La Tabla 4.3 muestra que todos los códigos sufren de degradación en la calidad de la solución a medida que la discretización se hace más basta (tamaño de elemento mayor), así como con el incremento en el paso de integración temporal. Sin embargo, note que un cambio de un orden de magnitud en el paso temporal afecta la solución, en todos los caso, en menos de un 10% en todos los casos para la misma discretización espacial. Sin embargo un cabio de cinco veces en el tamaño de la discretización espacial da lugar a cambios en la solución de hasta un 180% en algunos casos.

Para evaluar el impacto de la resolución espacial en la curvatura de la onda de despolarización (onda de activación), así como la propagación a lo largo y através dela dirección de la fibra, se evaluaron los tiempos de activación en el plano mostrado en la Figura 4.3. La Figura 4.5 muestra los tiempos de activación para la discretización espacial más fina (Δx=0.1mm) y más gruesa (Δx=0.5mm) y con Δt=0.005ms. La figura muestra que la implementación en GPU y el código FEniCS han capturado mejor la

conducción en la dirección transversal a la fibra para el caso de mallas bastas.

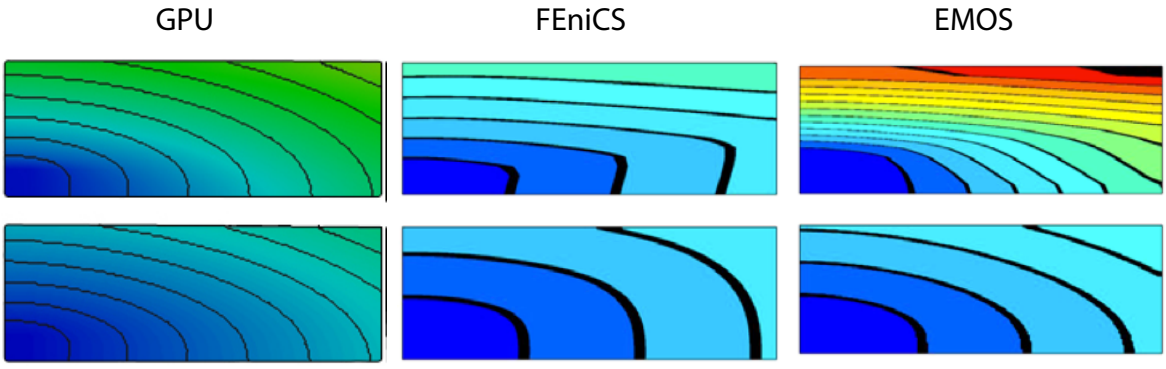


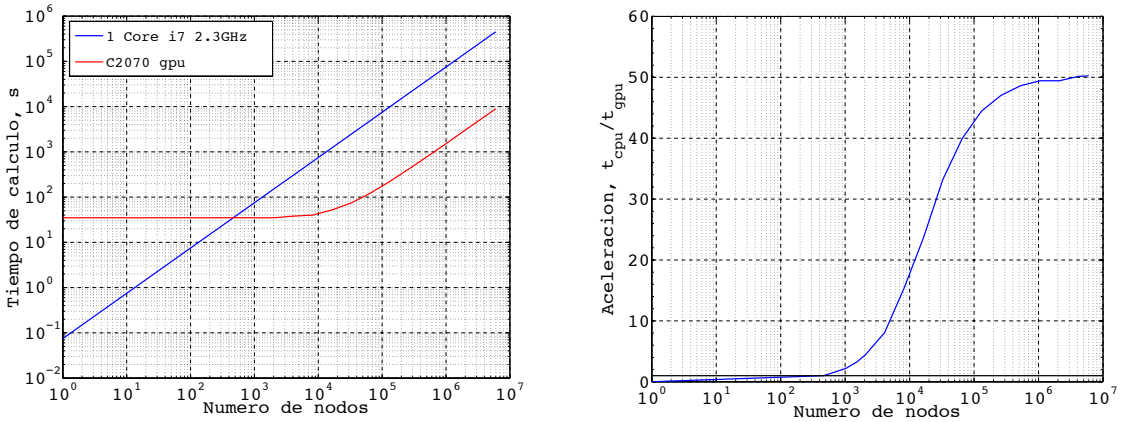
Figura 4.5. Tiempo de activación en el plano mostrado en la Figura 4.3 para los tres códigos considerados. Los paneles superior e inferior corresponden a los tiempos de activación para para Δx=0.5mm y Δt=0.005ms y Δx=0.1mm y Δt=0.005ms, respectivamente. El mapa de colores va desde el color azul oscuro (0ms) hasta el rojo (130ms) con los contornos a intervalos de 10ms.

4.3. Rendimiento de la implementación GPU

El rendimiento de la implementación en GPU se ha evaluado a dos niveles. El primer nivel consiste en el cómputo del modelo iónico únicamente a mediada que se incrementan los grados de libertad (número de nodos) del problema. Cabe destacar que la solución numérica de la actividad eléctrica del corazón implica problemas con más de un millón de grados de libertad. El segundo nivel en la evaluación de rendimiento comprendió el tiempo de computación en los problemas de propagación en 1D, 2D y 3D para estudiar el efecto del ancho de banda de la matriz en los tiempos de cómputo.

La Figura 4.6 muestra el rendimiento de la implementación GPU en la evaluación del modelo iónico con respecto al obtenido en un único núcleo de CPU (Core i7 a 2.3 GHz). La Figura 4.6a muestra el tiempo de calcular 1.0 s de actividad celular en la GPU y en un único CPU para diferente número de nodos del modelo. Tal y como se aprecia en la figura, el tiempo de cálculo de un hilo de ejecución (un solo nodo) en la GPU es aproximadamente 450 veces más lento que ejecutarlo en la CPU. Sin embargo, este tiempo de ejecución permanece constante hasta que el número de hilos alcanza 1024, momento en el cual empieza a incrementar hasta alcanzar una relación lineal con el número de nodos del problema. Por otro lado, el tiempo de ejecución en el CPU incrementa linealmente con el número de nodos mostrando un

performance superior a la GPU para un número de nodos menor a 450. Cuando el número de nodos en el modelo excede el millón (lo que representa un modelo de corazón completo), el rendimiento de la GPU es de 50x con respecto a la de la CPU simple, tal y como lo muestra la Figura 4.6b. Estos resultados indican que para calcular 1.0 s de actividad celular en el mismo tiempo que en una GPU es necesario una CPU con 50 núcleos.

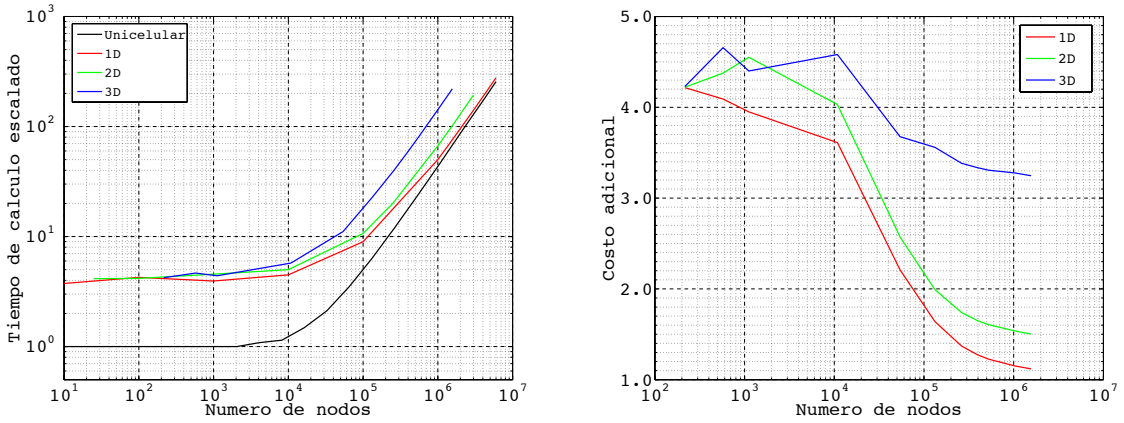


(a) (b)

Figura 4.6. Rendimiento en la evaluación del modelo iónico. a) Tiempo total de cálculo para la implementación GPU y un procesador CPU en función del número de nodos en la malla. b) aceleración de la implementación GPU con respecto a procesador CPU simple en función del número de nodos de la malla.

Para estudiar el rendimiento en el caso del problema de propagación del frente eléctrico, se consideraron problemas 1D, 2D y 3D con diferentes tamaño de malla entre 100 y 1.5×10^6 grados de libertad, pero manteniendo el tamaño del elemento fijo en 0.1 mm. La geometría 2D consistió en un cuadrado mallado con elementos cuadriláteros, mientras que la geometría 3D consistió en un cuboide mallado con hexaedros. De esta manera, se garantiza que el ancho de banda se mantiene fijo en los diferentes tipos de problemas: 3 para el problema 1D, 9 en el problema 2D y 27 en el problema 3D.

La Figura 4.7 muestra el rendimiento de la implementación GPU en la solución de 1.0 s de actividad eléctrica con un paso de integración temporal de 0.02ms. Los tiempos de cálculo han sido normalizados con respecto al tiempo empleado por un hilo de la GPU en resolver un segundo de actividad del modelo celular únicamente.



(a) (b)

Figura 4.7. Rendimiento en la resolución del modelo de propagación. a) Tiempo total de cálculo para la implementación GPU escalado con respecto al tiempo empleado en evaluar el modelo iónico por un solo hilo. b) Costo adicional empleado por el programa en resolver el modelo de propagación.

La Figura 4.7a demuestra que existe un coste adicional asociado a la operación matricial implicada en la ecuación (16). Sin embargo se observa que la tendencia de la evolución del tiempo de cálculo con el número de nodos del problema es la misma en los cuatro casos. Vemos que hasta no superar los 1000 grados de libertad, el tiempo empleado por la GPU es independiente del número de nodos considerados, incluso del ancho de banda del problema, en el caso del problema de propagación. Sin embargo, a medida que el número de nodos en el problema incrementa, el ancho de banda de la matriz $\mathbf{K}$ marca una diferencia en el coste computacional de la resolución del problema, con un mayor coste computacional para aquellas matrices con mayor ancho de banda. Note sin embargo, que ha medida que se incrementa el tamaño del problema, el coste relativo de resolver el problema de propagación con respecto al de resolver únicamente el modelo iónico en cada nodo se reduce hasta alcanzar un valor asintótico que depende de la dimensión del problema (1D, 2D, o 3D), así como del tipo de discretización empleada (ancho de banda de la matriz de rigidez). En este sentido, una discretización basada en elementos cuadriláteros en 2D y una de hexaedros en 3D representa siempre una cota inferior ya que son mallas que ofrecen el mínimo ancho de banda de la matriz de rigidez $\mathbf{K}$.

5 Conclusiones y trabajo futuro

La naturaleza multi-escala del problema de electrofisiología (constantes de tiempo de la cinética diferentes que van desde 0,1 a 500 ms) dificulta su solución numérica, que requieren resoluciones temporales y espaciales de 0,1 ms y 0,2 mm, respectivamente, para simulaciones precisas, dando lugar a modelos con millones grados de libertad que necesitan ser resueltos para mil pasos de tiempo. La solución de este problema requiere el uso de algoritmos con un mayor nivel de paralelismo en las plataformas multi-nucleo. En este sentido, las nuevas unidades programables de procesamiento gráfico (GPU) se han convertido en una alternativa debido a su altísima capacidad de paralelización y multiproceso gracias a la enorme cantidad de núcleos que es capaz de albergar en un mismo dispositivo.

En este trabajo se presenta los resultados obtenidos con una implementación de un programa de simulación de electrofisiología basado en el método de elementos finitos desarrollado íntegramente en CUDA. El software implementa un programa de solución explícita para el modelo de monodominio, utilizando la técnica de partición del operador y es capaz de manejar mallas en 1D, 2D y 3D. El modelo celular de ten Tussher y Panfilov (TP06) se ha considerado para modelar la célula aislada.

Los resultados obtenidos con una GPU NVIDIA C2090 en la simulación de 1.0 s de actividad de las células muestran que la GPU escala no linealmente con el número de nodos. Para problemas pequeños (entorno a los 1000 nodos), la GPU muestra un rendimiento constante e inferior al mostrado por un núcleo de una CPU (core i7 a 2.3 GHz). Sin embargo, a medida que aumenta el tamaño del problema, el rendimiento de la GPU supera al de una sola CPU, hasta alcanzar una aceleración de 50x. El rendimiento casi constante de la GPU observado para un bajo número de nodos se debe a que la carga no satura la capacidad de la GPU. Esto quiere decir que el modelo de tarjeta empleada en las pruebas es capaz de manejar hasta 1024 hilos simultáneos sin mostrar un detrimento en su rendimiento. Este resultado es por otro lado esperado ya que la arquitectura de la C2090 posee 512 núcleos de cálculo, pudiendo manejar hasta 1024 hilos de manera simultanea. A partir de los 1024 nodos y hasta los 80000 nodos el costo computacional se incrementa no linealmente y a partir de los 80000, la GPU escala linealmente con el número de nodos del problema considerado.

La clave a la hora de obtener un alto rendimiento en la GPU se encuentra sin embargo en cuan rápido un único hilo de la GPU sea capaz de integrar el modelo iónico. En este punto juegan un papel importante tanto la velocidad del reloj de cada uno de los núcleos de la GPU, como la estructura del código CUDA. En la estructura del código CUDA se encuentra la posible explicación del bajo rendimiento observado entre un único hilo de la GPU y la CPU (un rendimiento cerca de 450 veces menor). A nivel de programación, cada "Kernel" solo puede contener un número máximo de instrucciones que obligan a fraccionar el código, para evaluar el modelo iónico, en varias subrutinas, o "kernels", en lugar de un solo procedimiento. Esto se traduce en que, a la hora de evaluar el modelo iónico, el código debe invocar a varios "kernels" lo que implica el paso de datos y acceso a los mismo desde la memoria global del dispositivo incremento el tiempo de ejecución, ya que en este caso, el coste de acceder a los datos supera al de llevar a cabo las operaciones matemáticas. Está claro que estos resultados dependerán en gran medida del modelo iónico empleado y de la tarjeta GPU con la que se trabaje.

Cuando se considera el problema de propagación de la señal eléctrica, se observa la misma tendencia que para el caso unicelular. Sin embargo, se observa un incremento adicional en el coste computacional debido a las operaciones implicadas en el producto matricial. Cabe destacar que el incremento en el coste computacional es máximo para un bajo número de nodos para luego ir descendiendo hasta alcanzar un valor asintótico que es siempre mayor para aquellos casos en el que para el mismo número de nodos, la matriz de rigidez $\mathbf{K}$ presenta un mayor ancho de banda, ya que involucra un mayor número de operaciones matemáticas por iteración, así como una mayor cantidad de datos a ser mapeados e indexados dentro del dispositivo. A pesar de que las operaciones matriciales han sido ejecutadas con las librerías CUSP (Sparse Linear Algebra and Graph computations on CUDA), que optimizan dicha operación matemática, el tiempo de cálculo y por tanto el rendimiento obtenido dependen en gran medida de como se almacena la matriz de rigidez $\mathbf{K}$. En este sentido, el formato que se ha empleado en la implementación, ie., CSR, no resulta óptimo desde el punto de visto de la coalescencia de los datos en memoria lo que implica una pérdida en rendimiento de hasta cuatro veces con respecto a un tipo de almacenaje óptimo [21].

De los resultados obtenidos se puede concluir que en la solución del problema de electrofisiología cardiaca, la GPU ofrece una alternativa válida para la computación de alto rendimiento comparado con una CPU, o una arquitectura multi-núcleo, si se trata de problemas con un número de nodos por encima de 500000. Esta restricción es por otro lado cónsona con el problema bajo consideración ya que, un modelo anatómico del corazón posee varios millones de grados de libertad. Por otro lado, la implementación en GPU obedece a una consideración detallada no solo del problema, pero también de la arquitectura en la que se llevará a cabo si se quiere maximizar el rendimiento del código de simulación.

Como trabajo futuro se plantea:

- Delineamiento ("Profiling") de la implementación del modelo iónico en CUDA con la finalidad de identificar el número óptimo de sub-procedimientos ("kernel") a emplear, así como identificar aquellos procedimientos que consumen un mayor tiempo de computación.
- Integración completa de las librerías CUSP, incluyendo los algoritmos de resolución de sistema de ecuaciones lineales. De esta manera se podrá resolver el problema de forma implícita. Adicionalmente, con la completa incorporación de estas librerías se podrá acceder a diferentes modos de almacenaje SPARSE y estudiar su rendimiento en el problema de electrofisiología cardiaca.
- Incrementar la capacidad del código al manejo de mallas mixtas con elementos 1D, 2D y 3D simultáneamente, así como diferentes modelos iónicos dentro de un mismo modelo. Esto permitirá modelar el corazón junto a la red de Purkinje para llevar a cabo simulaciones más realistas.
- Implementar el código para trabajar en plataformas multi-GPU lo que permitiría abarcar problemas más grandes manteniendo la velocidad de cálculo siempre al óptimo.

6 Práctica clínica

La práctica clínica fue realizada en el hospital Clínic de Barcelona. Se podría dividir en dos partes. La primera más experimental, observación del comportamiento cardiaco de un cerdo. Y la segunda una operación de ablación sobre un paciente real. Las dos partes fueron llevadas acabo en la unidad de arritmias del Hospital Clinic por el grupo médico del que forma parte el Dr. Antonio Berruezo (persona de contacto).

La ablación cardíaca (o por radiofrecuencia) consiste en introducir por la ingle del paciente un catéter y hacerlo llegar a través de la arteria en el corazón para, una vez allí, quemar la vía accesoria o "cable" que genera el cortocircuito . Este tratamiento elimina la arritmia y permite que el paciente pueda llevar una vida absolutamente normal. Se trata de una técnica que requiere de una gran precisión, sobre todo en los casos de los niños que pesan menos de 15 kilos.

En la primera parte de la práctica clínica se atendió a un experimento sobre corazones infartados de cerdo antes y después de bloquear la rama derecha del Haz de His y estudiar el posicionamiento óptimo de un sistema de marcapasos. En este caso, se hizo uso de un sistema de visualización angiográfico (fluoroscopia) acoplado a un sistema CARTO 2D, para el guiado de los catéteres de ablación en el ventrículo del corazón infartado. En este experimento fue posible ver el efecto que un bloqueo de rama tiene sobre la señal electrocardiográfica y como los efectos de este bloqueo pueden minimizarse con la ayuda de una terapia de resincronización cardiaca.

En la segunda partes vimos la operación de un paciente, al que le estaban practicando una ablación auricular a nivel de la vena pulmonar. Aquí se tuvo la oportunidad de ver un sistema de navegación CARTO 3D en el que se indicaban los puntos de ablación así como los puntos donde se mide el electrograma de pared auricular, similar al mostrado en la Figura 6.1.

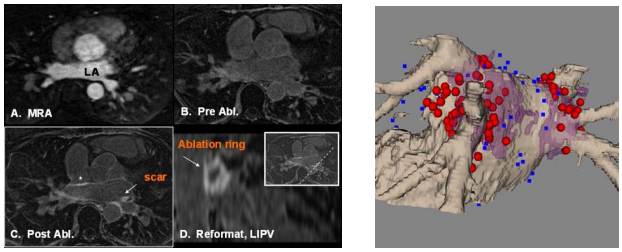


Figura 6.1. Imágenes del sistema CARTO 3D. La Figura de la deracha muestra una imagen de resonancia magnética en la que se superponen los puntos de ablasión

En el sistema CARTO 3D se aprovecha las imágenes de la resonancia magnética para reconstruir la geometría 3D de la aurícula y proveer, de ese modo, al médico con una imagen tridimensional que facilita la navegación con el catéter por la pared auricular identificando los diferentes puntos de ablación practicados (puntos rojos en la Figura 6.1). La experiencia fue particularmente interesante ya que ha permitido observar el uso de la tecnología de procesamiento de señales e imagen en asistir a un equipo médico en una intervención quirúrgica.

7 Bibliografía

[1] Katz AM. *Physiology of the Heart*. Lippincott Williams & Wilkins, 2005.

[2] Murphy JG, Lloyd MA. Mayo Clinic Cardiology. Mayo Clinic scientific press and inform Halthcare USA, Inc., 2007.

[3] Tung L. A bi-domain model for describing ischemic myocardial d-c potentia. PhD thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science, 1978.

[4] Sundnes J, Lines GT, Ca X, Nielsen BF, Mardal KA, Tveito A. Computing the Electrical Activity in the Heart. Springer-Verlag, 2006.

[5] Strang G. On the construction and comparison of difference schemes. SIAM J Num Anal, 5(3):506--517, 1968.

[6] Sundnes J, Lines GT, Tveito A. An operator splitting method for solving the bidomain equation coupled to a volume conductor model for the torso. Math Biosci, 194(2):233--248, 2005.

[7] Heidenreich EA, Ferrero JM, Doblare M, Rodriguez JF. Adaptive macro finite elements for the numerical solution of monodomain equations in cardiac electrophysiology. Annals Biomed Eng, 38(7):2331--2345, 2010.

[8] Reddy JN, Gartling DK. *The Finite Element Method in Heat Transfer and Fluid Dynamics*. CRC Press, 1994.

[9] Hughes TJR. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Englewwog Cliffs, NJ: Prentice Hall Inc., 672 pp., 1987.

[10] ten Tusscher K H W J, Noble D, Noble P J, Panfilov A V. A model for human ventricular tissue. Am J Physiol Heart Circ Physiol, 286(4):1573--1589, 2004.

[11] ten Tusscher K H W J, Panfilov A V. Alternans and spiral breakup in a human ventricular tissue model. Am J Physiol Heart Circ Physiol, 291(3):1088--1100, 2006.

[12] Luo C H, Rudy Y. A dynamic model of the cardiac ventricular action potential. I. Simulations of ionic currents and concentration changes. Circ Res, 74(6):1071--1096, 1994.

[13] Luo C H, Rudy Y. A dynamic model of the cardiac ventricular action potential. II. Afterdepolarizations, triggered activity, and potentiation. Circ Res, 74(6):1097--1113, 1994.

[14] Nickolls J, Buck I, Garland M, Skadron K. Scalable parallel programming with CUDA. Queue, 6(2):40–53, 2008

[15] NVIDIA Corporation. *NVIDIA CUDA Programming Guide*, June 2008. Version 2.0

[16] Kirk DB, Hwu WW. *Programming massively parallel processors. A hands-on approach*. Elsevier, 2010.

[17] Zienkiewicz OC, Taylor RL. *Finite Element Method. Vol. 1*, 752 pp. Elsevier Butterworth-Heinemann, Burlington, MA, 2005.

[18] *CUSP Libraries*. http://code.google.com/p/cusp-library/.

[19] *TP06 model source code*. http://www-binf.bio.uu.nl/khwjtuss/SourceCodes/

[20] Niederer SA, Kerfoot E, Benson AP, et al. Verification of cardiac tissue electrophysiology simulators using an N-version benchmark. Philos T Roy Soc A 369: 4331-4351, 2011.

[21] Bell N, Garland M. Efficient Matrix-Vector Multiplication on CUDA. NVIDIA Technical Report NVR-2008-004, 2008.